PATENT ABSTRACTS OF JAPAN

(11)Publication number : 05-100952
(43)Date of publication of application : 23.04.1993

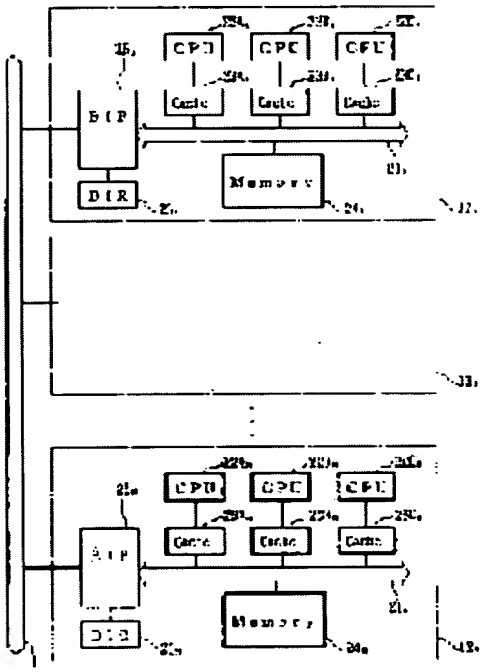
(51)Int.Cl. G06F 12/08
G06F 12/08
G06F 15/16

(21)Application number : 03-258819
(22)Date of filing : 07.10.1991

(71)Applicant : FUJI XEROX CO LTD
(72)Inventor : YANAGISAWA KATSUHIKO
KASUYA ATSUSHI

(54) DATA PROCESSOR

(57)Abstract:
PURPOSE: To provide the data processor having an economical and large scale multi-processor.
CONSTITUTION: This data processor is constituted of plural processor segments 12 constituted of plural processors 22, a memory 24, cache memories 23 which are placed at every processor 22 and hold a copy of a partial area of the memory 24, a memory bus 21 having a snoop mechanism and for coupling these cache memories 23 and the memory 24, and a bus interface part 25 connected to the memory bus 21, respectively, and a system bus 11 for connecting them. In the bus interface part 25 of every processor segment 12, a directory 26 for storing address information of data in the own processor segment 12 held in the cach memory 23 in another processor segment connected to the system bus 11 is arranged.



LEGAL STATUS

[Date of request for examination] 14.08.1997
[Date of sending the examiner's decision of rejection] 26.03.2002
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]
[Date of final disposal for application]
[Patent number]
[Date of registration]
[Number of appeal against examiner's decision of

rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（A）　(11)特許出願公開番号

特開平5－100952

(43)公開日　平成5年（1993）4月23日

(51)Int.Cl.[5]		識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ	12/08		Ｈ　7232－5B		
		３１０	Ｂ　7232－5B		
	15/16	３２０	Ｋ　8840－5L		

審査請求　未請求　請求項の数３（全 24 頁）

(21)出願番号　特願平3－258819

(22)出願日　平成3年（1991）10月7日

(71)出願人　000005496
富士ゼロツクス株式会社
東京都港区赤坂三丁目3番5号

(72)発明者　柳沢　克彦
埼玉県岩槻市府内3丁目7番1号　富士ゼロツクス株式会社岩槻事業所内

(72)発明者　粕谷　淳
埼玉県岩槻市府内3丁目7番1号　富士ゼロツクス株式会社岩槻事業所内

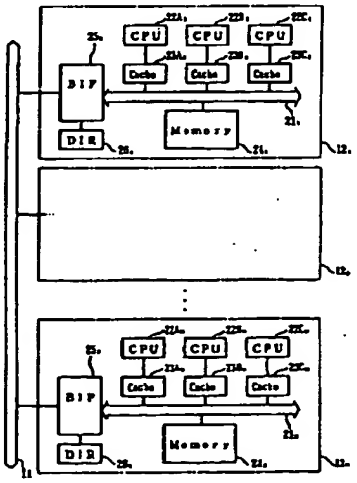
(74)代理人　弁理士　山内　梅雄

(54)【発明の名称】データ処理装置

(57)【要約】

【目的】　経済的で大規模なマルチ・プロセッサを有するデータ処理装置を提供する。

【構成】　データ処理装置を、複数のプロセッサ２２と、メモリ２４と、それぞれのプロセッサ２２ごとに配置されメモリ２４の一部領域のコピーを保持するキャッシュ・メモリ２３と、スヌープ機構を有しこれらキャッシュ・メモリ２３とメモリ２４を結合するメモリ・バス２１と、このメモリ・バス２１に接続されたバス・インターフェース部２５とからそれぞれ構成される複数のプロセッサ・セグメント１２と、これらを接続するシステム・バス１１とで構成する。プロセッサ・セグメント１２ごとのバス・インターフェース部２５には、システム・バス１２に接続された他のプロセッサ・セグメント１２内のキャッシュ・メモリ２３に保持されている自己のプロセッサ・セグメント１２内のデータのアドレス情報を記憶するディレクトリ２６が配置されている。



1

【特許請求の範囲】

【請求項1】　複数のプロセッサと、メモリと、それぞれのプロセッサごとに配置され前記メモリの一部領域のコピーを保持するキャッシュ・メモリと、スヌープ機構を有しこれらキャッシュ・メモリと前記メモリを結合するメモリ・バスと、このメモリ・バスに接続されたバス・インターフェース部とからそれぞれ構成される複数のプロセッサ・セグメントが、それぞれの前記バス・インターフェース部を介して共通のシステム・バスと接続されており、これらプロセッサ・セグメントごとのバス・インターフェース部には、前記システム・バスに接続された他のプロセッサ・セグメント内のキャッシュ・メモリに保持されている自己のプロセッサ・セグメント内のデータのアドレス情報を記憶するためのディレクトリが配置されていることを特徴とするデータ処理装置。

【請求項2】　複数のプロセッサと、メモリと、このメモリの一部領域のコピーを保持するキャッシュ・メモリと、スヌープ機構を有しこれら複数のプロセッサと、メモリおよびキャッシュ・メモリを結合するメモリ・バスと、このメモリ・バスに接続されたバス・インターフェース部とからそれぞれ構成される複数のプロセッサ・セグメントが、それぞれの前記バス・インターフェース部を介して共通のシステム・バスと接続されており、これらプロセッサ・セグメントごとのバス・インターフェース部には、前記システム・バスに接続された他のプロセッサ・セグメント内のキャッシュ・メモリに保持されている自己のプロセッサ・セグメント内のデータのアドレス情報を記憶するためのディレクトリが配置されていることを特徴とするデータ処理装置。

【請求項3】　複数のプロセッサと、メモリと、このメモリと接続されその一部領域のコピーを保持するキャッシュ・メモリと、スヌープ機構を有し前記複数のプロセッサと前記キャッシュ・メモリとを結合するメモリ・バスと、このメモリ・バスに接続されたバス・インターフェース部とからそれぞれ構成される複数のプロセッサ・セグメントが、それぞれの前記バス・インターフェース部を介して共通のシステム・バスと接続されており、これらプロセッサ・セグメントごとのバス・インターフェース部には、前記システム・バスに接続された他のプロセッサ・セグメント内のキャッシュ・メモリに保持されている自己のプロセッサ・セグメント内のデータのアドレス情報を記憶するためのディレクトリが配置されていることを特徴とするデータ処理装置。

【発明の詳細な説明】

【0001】

【産業上の利用分野】本発明はマルチ・プロセッサのキャッシュ上でのデータの不一致の発生を防ぐ機構を有するデータ処理装置に関する。

【0002】

【従来の技術】高性能のワークステーションのようにデータの高速処理を必要とする装置の分野では、複数のプロセッサを用いてデータ処理を行うようにしたマルチ・プロセッサ形式を採用する傾向にある。これは、プロセッサ自体の性能が向上しているものの、単独のプロセッサを用いてデータ処理を飛躍的に高速化することはかなり困難なことによるものである。複数のプロセッサを用いる場合には、各プロセッサとメモリとの間に高速のキャッシュ・メモリを配置して、プロセッサとメモリ間のバス（以下メモリ・バスという。）の使用頻度を下げる工夫が一般的に採られている。

【0003】このような複数のプロセッサにそれぞれ対応してキャッシュ・メモリが存在すると、各キャッシュ・メモリ間に格納されているデータの間に不一致が発生する可能性が生じる。こうした不具合を防止するために、従来から次の2つの方式が一般に知られている。

【0004】（1）スヌープ方式：各キャッシュ・メモリがメモリ・バス上のアクセスを監視する方式である。

【0005】（2）ディレクトリ方式：主記憶上のメモリ・ブロックごとにキャッシュ状態を記憶する方式である。

【0006】このうち（1）のスヌープ方式については、例えばCache Coherence Protocols:Evaluation Using a Multiprocessor Simulation Model, James Archibald & Jean-Laup Bare, ACM Transaction on Computer System. Vol4. NO. 4 November 1986, page 273-298. にいろいろなスヌープ方式についての解説と性能の比較に関する記述がある。また、日経エレクトロニクス 1988 年11月28日号P101～121 の“100MPS時代へ向け胎動を始めたマルチ・プロセッサ型ワークステーション”にも、今後のワークステーションの方向性として、このスヌープ方式のアーキテクチャについての説明が行われている。

【0007】また、（2）のディレクトリ方式については、例えばDierctory-Based CacheCoherence in Large Scale Multiprocessor, D. Chaiken, C. Fields, K. Karinhara, A. Agarwal IEEEE Computer June 1990 Page 49 ～58. に各種ディレクトリ構成を含んだ説明が行われている。

【0008】

【発明が解決しようとする課題】このうち（1）のスヌープ方式では、すべてのキャッシュ・メモリがバス上のすべてのアクセスを監視する。したがって、大規模なマルチ・プロセッサにはその適用が難しいという問題がある。このような問題を避けるためにはキャッシュ・メモリを2段以上に階層化するといったような方式が採られるが、こうするとプロセッサと主記憶装置の間での実際のデータ転送レートが低下してしまうという問題が発生する。

【0009】これに対して（2）のディレクトリ方式では、キャッシュ・メモリのブロックサイズごとに主記憶側にディレクトリ情報を保持する必要がある。このた

め、主記憶容量の増加とともに、この情報の保持に必要な記憶素子の容量が膨大になる。したがって、このディレクトリ方式を用いて実用的なシステムを作成するのは困難であり、実験あるいは研究用のシステムが作成されるにとどまっている。

【0010】そこで本発明の目的は、経済的で大規模なマルチ・プロセッサを有するデータ処理装置を提供することにある。

【0011】

【課題を解決するための手段】請求項1記載の発明では、データ処理装置を、複数のプロセッサと、メモリと、それぞれのプロセッサごとに配置され前記メモリの一部領域のコピーを保持するキャッシュ・メモリと、スヌープ機構を有しこれらキャッシュ・メモリと前記メモリを結合するメモリ・バスと、このメモリ・バスに接続されたバス・インターフェース部とからそれぞれ構成される複数のプロセッサ・セグメントと、これらのプロセッサ・セグメントをこれらのバス・インターフェース部を介して接続する共通のシステム・バスとで構成している。そして、これらプロセッサ・セグメントごとのバス・インターフェース部には、同一のシステム・バスに接続された他のプロセッサ・セグメント内のキャッシュ・メモリに保持されている自己のプロセッサ・セグメント内のデータのアドレス情報を記憶するためのディレクトリを配置している。

【0012】すなわち請求項1記載の発明では、複数のプロセッサごとにキャッシュ・メモリを備えたプロセッサ・セグメントにおけるスヌープ機構を持ったメモリ・バス間をシステム・バスで結合し、自己のプロセッサ・セグメント外にキャッシュされている自己のメモリ・バス上のデータを記憶するディレクトリを、自己のバス・インターフェース部に配置したものである。

【0013】請求項2記載の発明では、データ処理装置を、複数のプロセッサと、メモリと、このメモリの一部領域のコピーを保持するキャッシュ・メモリと、スヌープ機構を有しこれら複数のプロセッサと、メモリおよびキャッシュ・メモリを結合するメモリ・バスと、このメモリ・バスに接続されたバス・インターフェース部とからそれぞれ構成される複数のプロセッサ・セグメントと、これらのプロセッサ・セグメントをこれらのバス・インターフェース部を介して接続する共通のシステム・バスとで構成している。そして、これらプロセッサ・セグメントごとのバス・インターフェース部には、同一のシステム・バスに接続された他のプロセッサ・セグメント内のキャッシュ・メモリに保持されている自己のプロセッサ・セグメント内のデータのアドレス情報を記憶するためのディレクトリを配置している。

【0014】すなわち請求項2記載の発明では、複数のプロセッサと、メモリおよびキャッシュ・メモリを結合し、スヌープ機構を持ったメモリ・バス間をシステム・

バスで結合し、自己のプロセッサ・セグメント外にキャッシュされている自己のメモリ・バス上のデータを記憶するディレクトリを、自己のバス・インターフェース部に配置したものである。

【0015】請求項3記載の発明では、データ処理装置を、複数のプロセッサと、メモリと、このメモリと接続されその一部領域のコピーを保持するキャッシュ・メモリと、スヌープ機構を有しこれら複数のプロセッサとキャッシュ・メモリとを結合するメモリ・バスと、このメモリ・バスに接続されたバス・インターフェース部とからそれぞれ構成される複数のプロセッサ・セグメントと、これらのプロセッサ・セグメントをこれらのバス・インターフェース部を介して接続する共通のシステム・バスとで構成している。そして、これらプロセッサ・セグメントごとのバス・インターフェース部には、同一のシステム・バスに接続された他のプロセッサ・セグメント内のキャッシュ・メモリに保持されている自己のプロセッサ・セグメント内のデータのアドレス情報を記憶するためのディレクトリを配置している。

【0016】すなわち請求項3記載の発明では、複数のプロセッサとキャッシュ・メモリとを結合しスヌープ機構を持ったメモリ・バス間をシステム・バスで結合し、自己のプロセッサ・セグメント外にキャッシュされている自己のメモリ・バス上のデータを記憶するディレクトリを、自己のバス・インターフェース部に配置したものである。

【0017】

【実施例】以下実施例につき本発明を詳細に説明する。

【0018】図1は本実施例におけるデータ処理装置の構成を表わしたものである。このデータ処理装置は、システム・バス11と、これに接続されたプロセッサ・セグメント12_1、12_2、……12_nから構成されている。本実施例でそれぞれのプロセッサ・セグメント12_1、12_2、……12_nは、同一の構成をしているので、次に第1のプロセッサ・セグメント12_1についてその構成を代表的に説明する。なお、この明細書では特に対象を限定して説明する必要がある場合を除いてデータ処理装置の各構成部品の添字を省いた形で説明を行うことにする。

【0019】第1のプロセッサ・セグメント12_1のメモリ・バス21_1には、3つのCPU$22A_1$、$22B_1$、$22C_1$が対応するキャッシュ・メモリ$23A_1$、$23B_1$、$23C_1$を介して接続されている。メモリ・バス21_1にはプログラムやデータを格納したメモリ24_1も接続されている。各プロセッサ・セグメント12_1、12_2、……12_n上のメモリ24_1は、それぞれ別のアドレスを持っており、全メモリ24_1、24_2、……24_nは、単一のアドレス空間上にマッピングされるようになっている。

【0020】更に、このメモリ・バス21_1にはシステ

5

ム・バス１１と接続されたバス・インターフェース（ＢＩＦ）２５1 も接続されている。バス・インターフェース２５1 には、自己のメモリ・バス２１1 上のメモリ２４1 内のデータが外部のキャッシュ・メモリ内に保持されていることを記憶するためのディレクトリ２６1 が接続されている。

【００２１】さて、このような構成のデータ処理装置でメモリ・バス２１を介して行われるメモリ・アクセスは、通常のスヌープ方式によるバス・プロトコルが用いられる。（1）のスヌープ方式についての前記した先行

6

技術に示されるように、これについては各種のプロトコルを考えることができる。本実施例では、キャッシュ・メモリ２３の書き込み時に、他のキャッシュ・メモリ２３上のエントリを無効化する“Write Invaidate”方式を用いることにする。

【００２２】次の表1は、メモリ・バス２１で用いられる信号を示している。

【００２３】

【表１】

信号名	定義
ＡＤ〔63：0〕	アドレス，データ・マルチプレックス・バス
ＡＳ＊	アドレス・ストローブ
ＡＣＫ＊	アクノリッジ
Ｅｒｒ＊	エラー・アクノリッジ
Ｓｈｒ＊	シェア（データ が複数のキャッシュ・メモリ に保持されている）
ＯＷＮ＊	オーナ（データ がキャッシュ・メモリ 上で更新されたことを示す）
ＣＬＫ	クロック信号

【００２４】このようなバス信号の他に、各プロセッサ間でバスの使用権の調停を行うためのアービトレーション用の信号が必要であるが、これについての説明は省略する。なお、表１の信号名の箇所に示した“＊”は、負論理の信号であることを表わす記号である。また、“ＡＤ〔63：0〕”とは、アドレスおよびデータがマルチプレックスされた６４ビットのバスであることを示している。“ＡＳ＊”は、Ｃ２２あるいはキャッシュ・メモリ２３がＡＤ〔63：0〕上にアドレス情報を出力したことを示す信号である。この“ＡＳ＊”が出力されたときを、アドレス・フェーズと呼ぶ。このときＡＤ〔63：0〕上には次の表２に示すように、アクセスのための情報が出力されるものとする。

【００２５】

【表２】

マルチプレックス・バス	信号名	
ＡＤ〔31：0〕	ＡＤＲ〔31：0〕	３２ビットのアドレスが出力される
ＡＤ〔55：32〕	―――	未使用
ＡＤ〔59：56〕	ＴＹＰＥ〔3:0〕	アクセス・コマンドのタイプ
ＡＤ〔63：60〕	ＩＤ〔3:0〕	メモリ・バス上のデバイス識別コード

【００２６】“ＡＣＫ＊”は、アクセスが完了したことを知らせる信号であり、バス・スレーブから出力される。Ｅｒｒ＊は、アクセスが異常終了したことを知らせる信号である。ＡＣＫ＊とＥｒｒ＊が同時に出力される場合をＲ＆Ｒ（Relinguish andRetry）と呼ぶ。この場合、メモリ・バス２１の使用を管理する管理者としてのバス・マスタは、一旦バスの使用権を開放した後に再びアクセスをやり直す。この機構により、バス使用権の取り合いによるデッド・ロックを回避することができる。

【００２７】“Ｓｈｒ＊”は、複数のキャッシュ・メモリ２３上にアクセス中のデータが保持されている信号である。この信号は、全キャッシュ・メモリ２３からオープン・コレクタ出力によってドライブされる。メモリ・バス２１上のキャッシュ・メモリ２３は、バス上のアクセスを監視し、自分のキャッシュ・メモリ２３内に保持されているアクセスを検知した場合、この信号を出力するようになっている。

【００２８】“ＯＷＮ＊”は、アクセス中のデータが別のキャッシュ・メモリ２３上で更新されていることを示す信号である。この信号は、更新されたデータを保持するキャッシュ・メモリ２３から出力される。この場合、メモリ２４上のデータは最新のものではないので、このデータはキャッシュ・メモリから返される。ＣＬＫは、メモリ・バス２１上の同期クロックである。

7

【0029】次に、表2に示したアドレス・フェーズでの情報について説明する。ADR〔31：0〕は、アクセス対象となるアドレスを指定するフィールドである。TYPE〔3:0〕は、アクセスするタイプを指定するフィ

8

ールドであり、次の表3のようにデコードされる。ID〔3：0〕は、バス上のデバイスIDである。

【0030】

【表3】

TYPE〔3：0〕	アクセス・タイプ
0000	ライト　(WR)
0001	リード　(RD)
0010	コヒーレント・インバリデート (CI)
0011	コヒーレント・リード　(CR)
0100	コヒーレント・リードでインバリデート (CRI)
0101～1111	未使用

【0031】アクセス・タイプの説明

【0032】表3における“WR”は、CPU22とキャッシュ・メモリ23からメモリ24に対する書き込みを意味し、更新されたキャッシュ・エントリのリプレース時の書き戻し等に使用される。“RD”は、キャッシュの対象とならないメモリ・エリアへの読み出しを意味し、キャッシュ・メモリ23のスヌープ動作の対象外である。“CI”は、すべてのキャッシュ・メモリ23へのエントリの無効化要求であり、キャッシュ・メモリ23がこのアドレスのコピーを保持していた場合には、そのエントリが無効化される。

【0033】“CR”は、キャッシュ対象エリアへの読み出しを意味する。他のキャッシュ・メモリ23内にこのアドレスのデータが保持されている場合、そのキャッシュ・メモリ23はSHR＊を出力する。また、他のキャッシュ・メモリ23上でデータが更新された場合には、そのキャッシュ・メモリ23はSHR＊とOWN＊信号を出力して更新されたデータを返送する。この場合、OWN＊信号によって、メモリ24は読み出しの動作を行わない。

【0034】“CRI”は、読み出しとエントリの無効化を同時に行うアクセスを意味する。このアドレスを保持するキャッシュ・メモリ23は、そのエントリを無効化する。他のキャッシュ・メモリ23上でデータが更新された場合には、OWN＊信号を出力し、更新されたデータを返した後に、そのエントリを無効化する。

【0035】バス・シーケンスの説明

【0036】図2は、以上のようなメモリ・バスの構成による実際のバス・アクセスのシーケンスを表わしたものである。このうち同図（a）は通常のRDサイクル（シーケンス）を表わしている。このRDシーケンスでは、前記したバス・マスタがアドレス情報をADに出力し、AS＊をこれに応じて出力すると、同じく前記したバス・スレーブがデータを返してACK＊を出力するようになっている。

【0037】同図（b）は、WRサイクル（シーケンス）を表わしたものである。バス・マスタがAS＊を出力したと同時にアドレス情報がAD上に出力され、続いて書込データが出力される。バス・スレーブは書き込みが完了した時点でACKを返し、バス・マスタは次の書込データを出力する。

【0038】同図（c）は、CRサイクル（シーケンス）を表わしたものである。この図には他のキャッシュ・メモリ23からOWN＊信号が出力された場合を示してある。この場合、データはOWN＊信号を出力したキャッシュ・メモリ23から替えされることになる。

【0039】この図2に示したように、メモリ・バス21のアクセスは32バイト単位（64ビットバス＝8バイト×4バス転送）で行われるものとする。また、これに伴って、各キャッシュ・メモリ23のブロック・サイズは32バイトであるものとする。

【0040】キャッシュ・コンシステンシィについての説明

【0041】以上説明したバイト・アクセスを用いて、メモリ・バイト21上のキャッシュ・メモリ23は常に自分がストアしているキャッシュ・エントリが他のキャッシュ・メモリ23と共有のものであるかどうかを管理し、データの不一致が発生しないように制御を行う。

【000★】図3は、キャッシュ・エントリがCPUおよびメモリ・バスの動作に対して取るべき状態遷移を示したものである。この図の中で示した（1）～（14）の記号はキャッシュ・メモリ23の状態であり、これは次の表4のように定義される。

【0042】

【表4】

9　　10

状態	定義
Invalid	キャッシュ・エントリが無効
Clean Exclusive	データは更新されてなく、このキャッシュ・メモリだけにデータが保持されている
Clean Shered	データは更新されてなく、他のキャッシュ・メモリにもデータを共有している
Owned Exclusive	データは更新されており、このキャッシュ・メモリだけにデータが保持されている
Owned Shered	データは更新されており、他のキャッシュ・メモリもデータを共有している

【００４３】この図３内の矢印は、各状態からの遷移を表わしたものである。次の表５～表７は各遷移トリガを示している。

【００４４】

【表５】

番号	遷移トリガ	説明
(1)	ＣＰＵのリード	ＣＰＵのリードがミスヒットした場合、キャッシュ・メモリはメモリ・バスに対してＣＲを行う。このときＳＨＲ＊信号が出力されなかった場合には、エントリはClean Exclusive となる。
(2)	ＣＰＵのリード	ＣＰＵのリードがミスヒットし、キャッシュ・メモリはＣＲを実行する。このときＳＨＲ＊信号が出力された場合、エントリはClean Sheredとなる。
(3)	ＣＰＵのライト	ＣＰＵのライトがミスヒットした場合、キャッシュ・メモリはＣＲＩを行って、最新データを読み出すと同時に、他のキャッシュ・メモリに書き込まれた内容を無効にする。この後、データを更新してOwned Exclusive になる。
(4)	ＣＰＵのライト	ＣＰＵがClean Exclusive のエントリへのライトを行った場合、データを更新し、エントリはOwned Exclusive となる。
(5)	ＣＰＵのライト	ＣＰＵがClean Sheredのエントリへのライトを行った場合、キャッシュ・メモリはメモリ・バスに対してＣＩを行って、他のキャッシュ・メモリのコピーを無効にした後、データを更新しOwned Exclusive となる。

【００４５】

【表６】

11　　12

番号	遷移トリガ	説明
(6)	バス上のＣＲ	バス上のＣＲがClean Exclusive のエントリにヒットしたとき、キャッシュ・メモリはＳＨＲ＊を出力し、エントリはClean Sheredになる。
(7)	バス上のＣＲ	バス上のＣＲがClean Sheredのエントリにヒットしたとき、キャッシュ・メモリはＳＨＲ＊を出力する。状態は不変。
(8)	バス上のＣＲ	バス上のＣＲがOwned Exclusive のエントリにヒットしたとき、キャッシュ・メモリはＳＨＲ＊とＯＷＮ＊信号を出力して、キャッシュ・メモリ内のデータを返す。エントリはOwned Sheredになる。
(9)	バス上のＣＲ	バス上のＣＲがOwned Sheredのエントリにヒットしたとき、キャッシュ・メモリはＳＨＲ＊とＯＷＮ＊信号を出力し、キャッシュ・メモリ内のデータを返す状態は不変。
(10)	バス上のＣＲＩ	バス上のＣＲＩがClean Exclusive のエントリにヒットした場合、そのエントリはInvalid にされる。
(11)	バス上のＣＩ，ＣＲＩ	バス上のＣＩ，ＣＲＩがClean Sheredのエントリにヒット した場合、そのエントリはInvalid にされる。

【００４６】　　【表７】

番号	遷移トリガ	説明
(12)	バス上のＣＩ，ＣＲＩ	バス上のＣＩ，ＣＲＩがOwned Sheredのエントリにヒットした場合、そのエントリはInvalid にされる。ＣＲＩの場合には、ＯＷＮ＊信号を出力し、データをマスタに返した後、Invalid になる。
	コピー・バック	エントリが他のアドレスをキャッシュするために必要な場合、更新されたデータをメモリに書き戻し、エントリはInvalid となる。
(13)	ＣＰＵのライト	ＣＰＵがOwned Sheredのエントリにライトを行った場合、キャッシュ・メモリはメモリ・バスに対してＣＩを行って、他のキャッシュ・メモリのコピーを無効化した後、データを更新してOwned Exclusive となる。
(14)	バス上のＣＲＩ	バス上のＣＲＩがOwned Exclusive のエントリにヒットした場合、キャッシュ・メモリはＯＷＮ＊信号を出力して、データを返し、エントリをInvalid にする。
	コピー・バック	コピー・バック時は、更新されたデータをメモリに書き戻して、エントリはInvalid となる。

【００４７】以上の制御シーケンスによって、キャッシュ・メモリ２３上のデータが更新される場合、唯一のキャッシュ・メモリ２３にのみそのデータが存在することが保証され、キャッシュ・メモリ２３間のデータの不一致が避けられる。

【００４８】以上説明したスヌープ方式の技術は公知のものであり、すでに多くの実システムが存在している。したがって、以上の説明は次に述べるシステム・バス１１の動作の理解の手助けとして行ったものである。

【００４９】バス・インターフェースの説明

【００５０】各メモリ・バス２１上のキャッシュ・メモリ２４は、以上説明したように通常のスヌープ機構を持ったマルチ・プロセッサ・システムとして動作する。これに対して、バス・インターフェース２５（図１）は、メモリ・バス２１に対して１つのキャッシュ・メモリであるかのように振る舞いながら、全セグメントを単一空間のシェアド・メモリ・マルチ・プロセッサ・システムとして動作させる。

【００５１】ＣＰＵ２２は命令あるデータのアクセスを行うとき、これが自己のプロセッサ・セグメント内のキャッシュ・メモリ２３あるいはメモリ２４上に存在する場合には、これを読み出すことは当然である。これ以外の場合、各バス・インターフェース２５は、システム・バス１１を介して他のプロセッサ・セグメント１２内のバス・インターフェース２５に要求を出すことになる。この相手側のバス・インターフェース２５は、自分のメ

モリ・バス２１上のデータが他のプロセッサ・セグメント１２内のメモリ・バス２１上のキャッシュ・メモリ２３に保持されていることをディレクトリ２６上に記憶しておくことによって、キャッシュ・コンシステンシィを保つ。

【００５２】ディレクトリ構造の説明

【００５３】通常のディレクトリ方式では、メモリの全ブロックに１つずつディレクトリがある。本実施例のディレクトリ２６は、限られた数のエントリした持たない、いわゆるキャッシュ的な動作を行う。したがって、あるメモリ・バス２１から同時に外部のキャッシュ・メモリ２３に保持できるデータの数には制限がある。

【００５４】ここで、このディレクトリ２６が１０２４エントリ×４ウェイ（ｗａｙ）のセット・アソシエイティブ・キャッシュ構造をとることにする。先に説明したように、キャッシュ・メモリ２３のブロック・サイズを３２バイトとすると、表2に定めたＡＤＲ〔３１：０〕のアドレスは、ＡＤＲ〔４：０〕がブロック内バイトオフセットを指し、ＡＤＲ〔１４：５〕が１０２４のキャッシュ・ラインを選ぶのに用いられる。残りのＡＤＲ〔３１：１５〕がキャッシュ・メモリ２３内に保持されるタグに分類される。

【００５５】次の表8は、ディレクトリ２６内の１エントリに保持されているデータを示している。

【００５６】

【表８】

名称	定義
ＴＡＧ〔１６：０〕	エントリ内に保持されているブロックのアドレス・タグ（ＡＤＲ〔３１：１５〕の値）
ＳＩＤ〔３：０〕	このブロックをキャッシュしているバス・インターフェースの識別コード
Ｖａｌｉｄ	このエントリが有効かどうかを示すビット
Ｄｉｒｔｙ	このブロックが外部キャッシュ・メモリ上で更新されたことを示すビット
Ｍｕｌｔｉ	このブロックが複数のバス・インターフェースのキャッシュ・メモリに保持されていることを示すビット

【００５７】ここでＳＩＤ〔３：０〕は、各バス・インターフェース２５ごとにユニークに割り当てられる識別コードを保持するためのフィールドである。これは、先に示したメモリ・バス２１上のデバイス・コードとは別のものである。

【００５８】次にフィールドの値と定義について説明する。

【００５９】（イ）Valid ＝０のとき、このエントリは無効であり、残りのフィールドは意味を持たない（Invalid）。

【００６０】（ロ）Valid ＝１、Dirty ＝０、Multi ＝０のとき、このエントリで示されるデータ・ブロックが、ＳＩＤ〔３：０〕で示されるバス・インターフェース２５上のキャッシュ・メモリ２３に保持されている。キャッシュ・メモリ２３上のデータは更新されていない（Clean Single状態）。

【００６１】（ハ）Valid ＝１、Dirty ＝０、Multi ＝１のとき、このエントリのデータ・ブロックは、複数のバス・インターフェース２５上のキャッシュ・メモリ２３に保持されている。これは、同一のバス・インターフェース２５上の複数のキャッシュ・メモリ２３Ａ～２３Ｃではなく、あくまでも別のメモリ・バス２１上のキャッシュ・メモリ２３である。この状態で、キャッシュ・メモリ２３上のデータは更新されていない（CleanMultiple状態）。この場合には、ＳＩＤ〔３：０〕の情報は意味を持たない。

【００６２】（ニ）Valid ＝１、Dirty ＝１、Multi ＝０のとき、このエントリのデータはＳＩＤ〔３：０〕で示されるバス・インターフェース２５上のキャッシュ・メモリ２３に保持され、データはキャッシュ・メモリ２３上で更新されている（Dirty Single）。

【００６３】（ホ）Valid ＝１、Dirty ＝１、Multi ＝０のとき、データはＳＩＤ〔３：０〕のバス・インターフェース２５上のキャッシュ・メモリ２３で更新され、他のバス・インターフェース２５上のキャッシュ・メモリ２３にもコピーが保持されている（Dirty Multiple状態）。

【００６４】システム・バス信号の説明

【００６５】各バス・インターフェース２５間を接続するシステム・バス１１は、バス・マスタが発行した要求を処理するために、バス・スレーブが更に別のバス要求を発行する機能をもっている。次の表９に本実施例で用いられているシステム・バス信号を示す。

【００６６】

【表９】

15　　16

信号名	定義	出力	入力	タイプ
SYAD[63:0]	アドレス・データ・マルチプレクス・バス	マスタ/スレーブ	マスタ, スレーブ	TS
MAS＊	マスタ・アドレス・ストローブ	マスタ	スレーブ	TS
MACK＊	マスタ・アクノリッジ	スレーブ	マスタ	TS
MERR＊	マスタ・エラー・アクノリッジ	スレーブ	マスタ	TS
SAS＊	スレーブ・アドレス・ストローブ	スレーブ	2次スレーブ	TS
SACK＊	スレーブ・アクノリッジ	2次スレーブ	マスタ, スレーブ	TS
SERR＊	スレーブ・エラー・アクノリッジ	2次スレーブ	マスタ, スレーブ	TS
SAVD＊	スレーブ・アクノリッジ・バリッド	スレーブ	マスタ	TS
IBSY＊	インバリデーション・ビジー	スレーブ, 2次スレーブ	マスタ, スレーブ	OD
SBSY＊	システム・バス・ビジー	マスタ	他のBIF	TS
ARB[5:00]	アービトレーション・バス	バス・リクエスタ	バス・リクエスタ	OD
ABS＊	アービトレーション・ストローブ	バス・リクエスタ	他のBIF	OD
ABRQ＊	アービトレーション・リクエスト	スレーブ	バス・リクエスタ	TS

【0067】ここで“SYAD〔63：0〕”とは、64ビット構成のアドレスやデータを時分割で転送するためのマルチプレックス・バスを指す。“MAS＊”とは、システム・バス・マスタがSYAD〔63：0〕にアドレス情報を出して、マスタ・バス・トランザクションを開始したことを示す信号である。“MACK＊”とは、システム・バス・スレーブがマスタ・バス・トランザクションの完了を示すための信号である。“MERR＊”とは、システム・バス・スレーブがマスタにマスタ・バス・トランザクションのエラーの終了を通知するための信号である。

【0068】“SAS＊”とは、システム・バス・スレーブが、マスタ・バス・トランザクションの処理のために、SYAD〔63：0〕にアドレス情報を出力し、スレーブ・バス・トランザクションを開始したことを示す信号である。“SACK＊”とは、スレーブ・バス・トランザクションのアクセス先である2次スレーブがトランザクションの終了を示すための信号である。“SERR＊”とは、2次スレーブがスレーブ・バス・トランザクションのエラー終了を知らせる信号である。

【0069】“SAVD＊”とは、スレーブがマスタに対して、2次スレーブからのSACK＊およびSERR＊の組み合わせデータをもって、マスタ・バス・トランザクションの終了とすることを知らせる信号である。

“IBSY＊”とは、システム・バス11でのインバリデート・リクエストの処理中であることを知らせる信号である。各バス・インターフェース25からオープン・ドレイン・ゲートにてドライブされ、全バス・インターフェース25上での処理が終了した時点でネゲートされる。

【0070】“SBSY＊”とは、バス・マスタがバスを使用中であることを示す信号である。“ARB〔5：0〕”とは、アービトレーション・バスを示す。このバスは各バス・インターフェース25上のアービトレーション回路（後に図6で説明する。）を介してオープン・コレクタ・ゲートにてドライブされ、バス上の唯一のバス使用権者を選択する。

【0071】“ABS＊”とは、アービトレーション・ストローブを示す。ARB〔5：0〕を使用したアービトレーション・サイクルの実施中であることを示している。各バス・インターフェース25は、ABS＊が出力されていないことを検知した次のクロック・エッジからABS＊を出力して、アービトレーション・サイクルを開始することができる。“ABRQ＊”とは、アービトレーション・リクエストを示す。この信号は現在進行中のアービトレーション・サイクルを中断し、再度やり直しをリクエストするために用いられる。この信号を出力することができるのは、SBSY＊とABS＊とがともに出力されていることを検知した場合に限る。この信号を検知したアービトレーション参加者は、一旦、ABS＊をネゲートして、再度アービトレーション・サイクルをやり直す。

【0072】バス・アドレス情報の説明

【0073】MAS＊およびSAS＊が出力されたとき、SYAD〔63：0〕上には次の表10に示すアドレス情報が出力される。

【0074】

【表10】

17　　18

SYADバス	信号名	定義
SYAD[31:0]	ADR[31:0]	32ビットのアクセス・アドレス
SYAD[50:32]	—	未使用
SYAD[51]	MI	マルチ・インバリデート
SYAD[55:52]	TYPE[3:0]	コマンド・タイプ
SYAD[59:56]	TID[3:0]	ターゲット・ID
SYAD[63:60]	SID[3:0]	ソースID

【0075】ここで“ADR〔31：0〕”とは、32ビットのアクセス・アドレスが出力されることを示している。“MI”とは、インバリデートを要求する対象が、システム・バス11上の全バス・インターフェース25であることを示すビットである。このビットが有効になるのは、次に示す表11のSINおよびSTRIコマンドのときだけである。TYPE〔3：0〕は、システム・バス11上のアクセス・タイプを示している。タイプは、次の表11のようにデコードされる。

【0076】

【表11】

TYPE〔3：0〕	タイプ
0000	ライト（SWR）
0001	リード（SRD）
0010	アクセス・リクエスト（SA）
0011	コヒーレント・リード（SCR）
0100	コヒーレント・リードかつアクセス・リクエスト（SCRA）
0110	インバリデート（SIN）
0111	ターゲット・リード（STR）
1000	ターゲット・リード かつインバリデート（STRI）
1001～1111	未使用

【0077】TID〔3：0〕とは、特定のバス・インターフェース25に対して、アクセスを指定するフィールドをいう。このフィールドが有効なアクセスは、SIN、STRおよびSTRIのみである。ただし、SINとSTRIアクセスで、MIビットが“1”の場合には、全バス・インターフェース25に対するインバリデート要求となる。SID〔3：0〕とは、バス・トランザクションを開始したバス・インターフェース25のIDを出力することを示している。

【0078】<u>アクセス・タイプの説明</u>

【0079】表11に示したアクセス・タイプは次のように定義される。

【0080】① “SWR”は、システム・バス11経由で、他のメモリ・バス21上のメモリ24へデータを書き込む要求である。

【0081】② “SRD”は、システム・バス11経由で、他のメモリ・バス21上のメモリ24へデータを読み出す要求である。

【0082】③ “SAR”は、他のメモリ・バス21上のデータへの書き込み権の要求である。

【0083】④ “SCR”は、外部メモリ・バス21上のデータに対するコヒーレント・リードを要求するリクエストである。キャッシュ・メモリ23上でデータが更新された場合には、最新のデータが返されなければならない。

【0084】⑤ “SCRA”は、外部メモリ・バス21上、データへのコヒーレント・リードと書き込み権を要求するものである。データがキャッシュ・メモリ23上で更新されていた場合、その最新のデータが返される。このとき、この更新されたデータについて、更新される前のデータのコピーを保持するすべてのキャッシュ・メモリ23の内容は、最新のデータとは異なることになるので無効化されなければならない。

【0085】⑥ “SIN”は、外部メモリ・バス21上のキャッシュ・メモリ23へのインバリデートを要求するものである。MIビットが“0”の場合、TID

〔3：0〕で指定されるバス・インターフェース25上のキャッシュ・メモリ23のみが対象となる。MIビットが“1”の場合、全てのバス・インターフェース25がインバリデートの処理を行う。

【0086】⑦“STR”は、外部メモリ・バス21上のキャッシュ・メモリ23からデータを読み出す要求である。TID〔3：0〕で指定されるバス・インターフェース25上のキャッシュ・メモリ23が読み出しの対象である。

【0087】⑧“STRI”は、STRとSINを同時に行う要求である。MIビットが“0”の場合、TID〔3：0〕のバス・インターフェース25のみが対象となる。MIビットが“1”の場合、TID〔3：0〕のバス・インターフェース25上のキャッシュ・メモリ23が読み出しの対象であり、他のバス・インターフェース25もインバリデート処理を行う。

【0088】以上のアクセス・タイプのうち⑥～⑧のみがスレーブ・バス・トランザクションに用いられる。

【0089】システム・バス・トランザクション・シーケンスの説明

【0090】図4および図5は、以上のシステム・バス信号を用いて行われるバス・トランザクション・シーケンスを説明するためのものである。バス・トランザクション・シーケンスは、次の規則に従って動作する。

【0091】①マスタ・バス・トランザクションは、MAS＊の出力で開始され、MACK＊（またはMERR＊）にて終了する（図4（a））。

【0092】②SWR・マスタ・トランザクションでの書き込みデータは、アドレス・フェーズに引続き連続して転送され、MACK＊によるハンドシェークは行わない。MACK＊は全処理の終了時t_1に返される（図4（b））。

【0093】③SRD、SCR、SCRA、STRおよびSTRIのトランザクションにおけるスレーブ（または2次スレーブ）からの読み出しデータについては、MACK＊（またはSACK＊）によるハンドシェークが行われる。バス転送は32バイトのみであるため、4回の転送完了にてトランザクションが終了する（図4（c））。

【0094】④マスタ・トランザクションの処理のため、スレーブはSAS＊を用いて、スレーブ・トランザクションを開始することができる。スレーブ・トランザクションは、SAS＊の出力で開始され、SACK＊（またはSERR＊）にて終了する（図4（d））。

【0095】⑤SINおよびSTRIのアクセス時に、MIが“1”の場合、バス・マスタを除いたすべてのバス・インターフェース25がインバリデートの対象となる。バス・インターフェース25はこの条件を検知したとき、直ちにIBSY＊を出力し、各メモリ・バス21上でCI（コヒーレント・インバリデート）を実行する。マスタはIBSY＊のネゲートを待ってから、SBSY＊をネゲートし、バスを開放する（図5（e））。

【0096】⑥スレーブ・トランザクションを発行したスレーブは、SAVD＊を出力することによって、2次スレーブからのSACK＊（またはSERR＊）をもって、MACK＊（MERR＊）とすることができる。これによって2次スレーブからの応答をマスタに直接引き渡すことができる（図5（f））。なお、図5（f）における時刻t_3は、マスタ・トランザクションの完了の時刻を表わしている。

【0097】以上のシーケンスに従って、各バス・インターフェース25はシステム・バス11上でのデータ転送を行う。

【0098】アービトレーション・シーケンスの説明

【0099】各バス・インターフェース25は、メモリ・バス21上のアクセスによってシステム・バス11によるアクセスが必要になったとき、バス・アービトレーションを行う。本実施例でのアービトレーションは、SBSY＊、ABS＊、ARB〔5：0〕およびABRQ＊で行われる。以下アービトレーションの動作について説明するが、これらは本発明での特別な方式ではなく、公知の技術であり、ほじで説明するシステム全体の理解を容易にするためのものである。

【0100】バス使用権の必要なバス・インターフェース25は、ABS＊がネゲートされたことを検知後、ARB〔5：0〕にアービトレーション・コードを出力し、ABS＊をアサートしてアービトレーションに参加する。ARB〔5：0〕は、上位2ビットがプライオリティであり、下位4ビットに各バス・インターフェース25のIDが出力されるようになっている。これらはアービトレーション回路に入力される。

【0101】図6は、アービトレーション回路の構成を表わしたものである。アービトレーション回路41は、各位に対応したビットID〔0〕～ID〔3〕、PR〔0〕、PR〔1〕を入力する2入力オアゲート42_1～42_6および43_1～43_6を備えている。各アンドゲート43_1～43_6の出力端子にはオープン・コレクタ・ゲート44_1～44_6の入力側が配置されており、その出力側はインバータ45_1～45_6を介して対応する2入力オアゲート42_1～42_6のもう一方の入力端子に接続されている。また、このアービトレーション回路41では、各アンドゲート43_1～43_6にイネーブル信号が入力される他、アンドゲート43_5には42_6の出力が、またアンドゲート43_4には42_6および42_5の出力が、更にアンドゲート43_3には42_6、42_5および42_4の出力がそれぞれ入力されるようになっている。アンドゲート43_2および43_1についても同様に入力が増加している。そして、更に他のアンドゲート46には、イネーブル信号と42_1、42_2

～４２，の各出力信号が入力され、ＷＩＮ信号が出力されるようになっている。

【０１０２】この図６に示したアービトレーション回路４１では、上位ビットより自己出力とバス・データを比較し、データが不一致のときにはそのビットより下位の出力を禁止するようになっている。バスはオープン・コレクタでドライブされるため、常にバス上では“０”出力が優先し、ＡＲＢ〔５：０〕＝“０００００”が最も高いプライオリティを持つ。アービトレーションは、バスが安定状態に達するまで待ち、最終的に、この図６に示したアービトレーション回路４１にＷＩＮ信号を得たバス・インターフェース２５がバスの使用権を持つことになる。

【０１０３】ＡＲＢ〔５：４〕の２ビットは、プライオリティで、４レベルのプライオリティをサポートする。最高プライオリティＡＲＢ〔５：４〕＝“００”は、後に説明するディレクトリ・エントリのインバリデート処理のために用いられる。ＡＢＲＱ＊信号は、システム・バス１１のトランザクション処理中に、このインバリデート要求が発生したとき、現在決定している次のバス使用者を無効化し、再度アービトレーション・サイクルをやり直すために用いられる。

【０１０４】図７は、一連のアービトレーション・シーケンスを表わしたものである。サイクル“０”で、前のバス使用者がＳＢＳＹ＊を開放し、アービトレーションの勝者（ＷＩＮ０）がＳＢＳＹ＊をサイクル“１”より出力してバスを使用し始める。これと同時に、ＷＩＮ０はＡＢＳ＊およびＡＲＢ〔５：０〕をネゲートする。サイクル“２”にて、バス要求を持つバス・インターフェース２５は、ＡＢＳ＊のネゲートを検知し、ＡＢＳ＊および自己のアービトレーション値を出力して、次のアービトレーション・サイクルを開始する。

【０１０５】サイクル“４”にて、勝者（ＷＩＮ１）が決定するが、サイクル“５”でアービトレーションの再リクエスト（ＡＢＲＱ＊）が発生し、全バス・インターフェース２５は、ＡＢＳ＊をネゲートし、再びアービトレーションをやり直す。サイクル“９”で新しい勝者（ＷＩＮ２）が決定し、サイクル“１１”からバスを使用し始める。

【０１０６】本実施例にこのアービトレーション方式を用いた理由は、（ｉ）中央アービタのような特別な機構が必要なく、システムの拡張性が高い他、（ｉｉ）複数のプライオリティ・レベルが取り扱えるため、緊急の処理に対応可能である等による。もちろん、この方式を用いる必要は必ずしも存在せず、例えば通常用いられている各バス・ユーザごとの専用線と中央アービタによる方式でも構わない。

【０１０７】メモリ・バスに対するバス・インターフェースの動作

【０１０８】バス・インターフェース２５は、以上説明したバスの動作に従って、キャッシュ・メモリ２３間の記憶内容の不一致が発生しないように制御を行う。まず、自分のメモリ・バス２１上のトランザクションに対して、どのように動作するかについて説明を行う。

【０１０９】表３で示したメモリ・バス２１上のトランザクションに対して、バス・インターフェース２５が動作する必要のあるアドレス条件は、次の表１２のようになる。これ以外のアドレスは、ＣＰＵ２２、キャッシュ・メモリ２３およびメモリ２４間のみで解決する。

【０１１０】

【表１２】

条件	説明
External Clean (EC)	アクセス・アドレスが外部のメモリ・バスへのものであり、自己メモリ・バス上のキャッシュ・メモリが書き込み権を持っていない。（ＯＷＮ＊が出力されなかった）
External Dirty (ED)	アクセス・アドレスが外部のメモリ・バスへのものであり、自己メモリ・バス上のキャッシュ・メモリが書き込み権を持っている。（ＯＷＮ＊が出力された）
Hit Clean (HC)	アクセス・アドレスがディレクトリに登録されており、かつDirtyビットはセットされていない。（つまり、書き込み権は外部キャッシュ・メモリに渡されていない）
Hit Dirty (HD)	アクセス・アドレスがディレクトリに登録されており、かつDirtyビットがセットされている。（書き込み権が外部キャッシュ・メモリに渡されている）

【０１１１】表３のメモリ・バス・トランザクションと、表１２のアドレス条件に対して、バス・インターフェース２５は次の表１３および表１４に示す動作を行う。

【０１１２】

【表１３】

23　　24

メモリ・バス アクセス	アドレス条件	バス・インターフェース動作
WR	EC	システム・バスへSWR要求
	ED HC HD	発生しない 発生しない 発生しない
RD	EC	システム・バスへのSRD要求
	ED HC HD	発生しない 発生しない 発生しない
CI	EC	システム・バスへのSA要求
	ED	システム・バスへのSA要求
	HC	システム・バスへのSIN要求
	HD	システム・バスへのSIN要求

【０１１３】　　【表１４】

メモリ・バス アクセス	アドレス条件	バス・インターフェース動作
CR	EC	メモリ・バス・SHR#出力、システム・バスへのSCR 要求
	ED	メモリ・バス・SHR#出力後、何もしない(ローカル・キャッシュ が データ を 返す)
	HC	メモリ・バス・SHR#出力後、何もしない(メモリが データ を 返す)
	HD	メモリ・バス・SHR#、[illegible]#出力、システム・バスへのSTR 要求
CRI	EC	システム・バスへのSCRA要求
	ED	何もしない(ローカルキャッシュが データ を 返し、書き込み権が 移動)
	HC	システム・バスへのSIN要求
	HD	メモリ・バス OWN#出力、システム・バスへのSTRI要求

【０１１４】以上のように振る舞うことで、バス・インターフェース２５はメモリ・バス２１上からは１つのキャッシュ・メモリ２３のように見える。

【０１１５】バス・インターフェースのシステム・バスに対する動作

【０１１６】表１３および表１４のような動作によって、各バス・インターフェース２５はシステム・バス１１上に表１１に示すタイプのトランザクションを発生させる。これに対して、要求を受ける側のバス・インターフェース２５上のディレクトリ条件は、次の表１５のものが考えられる。

【０１１７】

【表１５】

25　　26

条件	説明
Miss Bit (M B)	要求アドレスのエントリは、スレーブ側のディレクトリ内に存在しない。
Clean Single (C S)	要求アドレスのエントリが、スレーブ側のディレクトリ内に存在し、(Dirty, Multi) = (0, 0)
Clean Multi (C M)	要求アドレスのエントリが、スレーブ側のディレクトリ内に存在し、(Dirty, Multi) = (0, 1)
Dirty Single (D S)	要求アドレスのエントリが、スレーブ側のディレクトリ内に存在し、(Dirty, Multi) = (1, 0)
Dirty Multi (D M)	要求アドレスのエントリが、スレーブ側のディレクトリ内に存在し、(Dirty, Multi) = (1, 1)

【０１１８】ここで、表１１のトランザクション・タイプのうち、ＳＩＮ、ＳＴＲおよびＳＴＲＩは、自己メモリ・バス２１のエリアではなく、アドレス・フェーズのＴＩＤおよびＭＩフィールドによって動作が定まる。次の表１６はこれを表わしたものである。

【０１１９】

【表１６】

アクセス	自己ＴＩＤ	ＭＩ	動作
ＳＩＮ	Ｙｅｓ	－	メモリ・バスへＣＩ実行
	Ｎｏ	0	何もしない
	Ｎｏ	1	メモリ・バスへＣＩ実行
ＳＴＲ	Ｙｅｓ	－	メモリ・バスへＣＲ実行
	Ｎｏ	0	何もしない
	Ｎｏ	1	発生しない
ＳＴＲＩ	Ｙｅｓ	－	メモリ・バスへのＣＲＩ実行
	Ｎｏ	0	何もしない
	Ｎｏ	1	メモリ・バスへのＣＩ実行

【０１２０】それ以外のアクセス・タイプについては、自己メモリ・バス２１のアドレスに対し、表１４のディレクトリ条件に対して、次の表１７および表１８のように動作する。

【０１２１】

【表１７】

27　　28

アクセスタイプ	ディレクトリ条件	動作
SWR	MH	自己メモリ・バスへのWR実行
	CS CM	発生しない 発生しない
	DS	自己メモリ・バスへのWR実行、ディレクトリ・エントリの無効化
	DM	自己メモリ・バスへのWR実行、Dirty ビット・クリア
SRD	MH	自己メモリ・バスへのRD実行
	CS CM DS DM	発生しない 発生しない 発生しない 発生しない
SA	MH	発生しない
	CS	メモリ・バスへのCI，(D，M)＝(1，0)
	CM	システム・バスへのSIN要求(MI＝1，スレーブ) メモリ・バスのCI，(D，M)＝(1，0)
	DS	発生しない
	DM	システム・バスへのSIN要求(MI＝1) メモリ・バスのCI，(D，M)＝(1，0)

【0122】　　【表18】

アクセスタイプ	ディレクトリ条件	動作
SCR	MH	新エントリ作成、メモリ・バスのCR実行、 (D，M)＝(0，0)
	CS	メモリ・バスへのCR実行、 (D，M)＝(0，1)／(0，0)　(##)
	CM	メモリ・バスへのCR実行
	DS	システム・バスへのSTR実行、 (D，M)＝(1，1)
	DM	システム・バスへのSTR実行
SCRA	MH	新エントリの作成、メモリ・バスへのCRI実行、 (D，M)＝(1，0)
	CS	システム・バスへのSIN要求、メモリ・バスへの CRI実行、(D，M)＝(1，0)　(#)
	CM	システム・バスへのSIN要求(MI＝1)、 メモリ・バスへのCRI実行、(D，M)＝(1，0)
	DS	システム・バスへのSTRI要求、SAVD*出力、 メモリ・バスへのCI実行、(D，M)＝(1，0)
	DM	システム・バスへのSTRI要求(MI＝1)、 SAVD*出力、メモリ・バスCI実行、 (D，M)＝(1，0)

【0123】表18において、（#）の箇所については、システム・バス11へのSINは、SIDと、エントリ内のIDが一致した場合には不要である。また、（##）の箇所については、SIDとエントリ内のIDが一致した場合には、Multiビットはセットしない。

【0124】ところで、表17および表18においては、新エントリを作成する場合、4ウェイ・セット・アソシエイティブの構成をとる。ディレクトリには、空がない場合が発生する。この場合には、4つのエントリか

ら1つを選び、そのエントリを無効化する必要がある。なお、選択にはランダム、ＬＩＦＯ（後入れ先出し）、ＬＲＵ（日本語名あるいはフルネーム？　？　？　）等の各種の方法を採ることができる。エントリの無効化の操作は、バス・インターフェース２５内に設けられたインバリデート・ペンディング・バッファ（ＩＰＢ）を用いて行われる。以下にその処理手順を示す。

【０１２５】（ａ）選ばれたエントリの内容が、インバリデート・ペンディング・バッファにコピーされる。

【０１２６】（ｂ）インバリデート・ペンディング・バッファのインバリデートは、アービトレーションの最高プライオリティを持つ。インバリデート・ペンディング・バッファにエントリがコピーされた場合、ＡＢＲＱ＊を用いてアービトレーションを再度行い、確実に現在処理中の次のバス使用権がインバリデート・ペンディング・バッファ処理に渡るようにする。

【０１２７】（ｃ）インバリデート・ペンディング・バッファ上のエントリは、メモリ・バス２１上のトランザクションをスヌープする機能を持つ。メモリ・バス２１上のＣＲまたはＣＲＩがエントリにヒットしてＤヒットが“1”の場合には、バスへＳＨＲ＊およびＯＷＮ＊を出力し、メモリが応答するのを禁止した後、先に説明したＲ＆Ｒアクノリッジを返し、アクセスのリライトを要求する。この操作は、インバリデートが終了するまで行

われ、インバリデート・ペンディング・バッファ上のエントリが無効になった後は、通常のメモリ・バス上のメモリ・アクセスとなる。

【０１２８】（ｄ）インバリデート・ペンディング・バッファ上のエントリのDirtyが“0”の場合には、システム・バス１１に対してＳＩＮ要求を出力する。Dirtyが“1”の場合、システム・バス１１にＳＴＲＩ要求を出し、更新されたデータを読み取り、メモリ・バス２１上のメモリ２４にWRで書き戻す。

【０１２９】以上により、新しいエントリの作成のための旧エントリは、次のシステム・バス・トランザクションで確実にインバリデートされ、それぞれのキャッシュ・メモリ２３間でのデータの不一致等の不都合は生じない。

【０１３０】これまで、バス・インターフェース２５の動作を中心に説明を行った。次にディレクトリの状態遷移について説明する。

【０１３１】ディレクトリ状態遷移の説明

【０１３２】図８は、ディレクトリの１エントリの状態遷移を表わしたものである。この図に示した各矢印での動作を次の表１８～表２０に示す。

【０１３３】

【表１９】

番号	トリガ	説明
(1)	ＳＣＲ	システム・バスからのＳＲが、ディレクトリに存在しなかった場合、新しいエントリを作り、Clean Singleとし、メモリからのデータをマスタに返す。
(2)	ＳＣＲＡ	システム・バスからのＳＣＲＡがディレクトリに存在しなかった場合新しいエントリを作り、Dirty Singleとし、メモリ・バスへＣＲＩを行ない、データをマスタに返す。
(3)	ＳＣＲ	システム・バスからのＳＣＲＡがClean Singleで存在し、ＳＩＤがエントリ内のＩＤと一致した場合、Clean Singleのままとし、メモリ・バス上の別のキャッシュからのＣＲ時に発生する。
(4)	ＳＣＲ	システム・バスからのＳＣＲがClean Single存在し、ＳＩＤが不一致の場合、Clean Multi とし、メモリ・バスＣＲのデータをマスタに返す。
(5)	ＳＣＲ	システム・バスからのＳＣＲが、Dirty Singleで存在した場合（ＩＤは一致し得ない）、Dirty Multi としシステム・バスへ、ＳＴＲのスレーブ・トランザクションを要求し、ＳＡＶＤ＊を出力して、2次スレーブからのデータをマスタに渡す。

【０１３４】

【表２０】

31　　32

番号	トリガ	説明
(6)	SCR	システム・バスからのSCRがClean Multi で存在した場合エントリは、Clean Multi のままで、メモリ・バスCRのデータをマスタに返す。
(7)	SCR	システム・バスからのSCRが、Dirty Multi で存在した場合エントリは、Dirty Multi のままで、システム・バスへSTRのスレーブ・トランザクションを要求し、SAVD*を出力して2次スレーブからのデータをマスタに渡す。
(8)	WR	システム・バスからのWAが、Dirty Multi で存在した場合、エントリをClean Multi とし、データをメモリへWAで書きもどす。
(9)	SA	システム・バスのSAが、Clean Singleで存在した場合、メモリ・バスにCIを行ない、エントリをDirty Singleとする。
	SCRA	システム・バスのSCRAがClean Singleで存在した場合、IDが不一致の場合、SINスレーブ・トランザクションを行ない、エントリをDirty Singleとしメモリ・バスCRIデータをマスタに返す。
(10)	SA	システム・バスのSAがClean Multi で存在した場合、MI＝1で、SINスレーブ・トランザクションを要求し、メモリ・バスにCIを行ない、エントリをDirty Singleとする。
	SCRA	システム・バスのSCRAがClean Multi で存在した場合、MI＝1で、STRIスレーブ・トランザクションを発行し、SAVD*を出力して、2次スレーブからのデータをマスタに渡し、メモリ・バスにCIを行ない、エントリをDirty Singleとする。

【0135】　　【表21】

番号	トリガ	説明
(11)	SA	システム・バスのSAが、Dirty Multi で存在した場合、MI＝1で、SINスレーブ・トランザクションを行ない、メモリ・バスにCIを行ない、エントリをDirty Singleとする。
	SCRA	システム・バスのSCRAがDirty Multi で存在した場合、MI＝1で、STRIスレーブ・トランザクションを発行し、SAVD＊を出力して、2次スレーブからのデータをマスタに渡す。メモリ・バスにCIを行ない、エントリをDirty Singleとする。
(12)	メモリ・バス CI, CRI	メモリ・バスのCI、CRIがClean Singleエントリにヒットした場合、システム・バスSINを行ない、エントリを無効化する。
	ディレクトリ リプレースメント	新しいディレクトリ・エントリのために、Clean Singleエントリを無効化する場合、システム・バスSINを行う。
(13)	メモリ・バス CI, CRI	メモリ・バスのCI、CRIがClean Multi エントリにヒットした場合、MI＝1でシステム・バスSINを行ない、エントリを無効化する。
	ディレクトリ リプレースメント	Clean Multi エントリを無効化する場合MI＝1でシステム・バスSINを行う。
(14)	メモリ・バス CI	メモリ・バスのCIが、Dirty Multi エントリにヒットした場合、MI＝1でシステム・バスSINを行ない、エントリを無効化する。
	メモリ・バス CRI	メモリ・バスのCRIが、Dirty Multi エントリにヒットした場合、OWN＊を出力して、メモリの応答を禁止し、MI＝1でシステム・バスにSTRIを行ない、データをメモリ・バスに返した後、エントリを無効とする。
	ディレクトリ リプレースメント	Dirty Multi のエントリを無効化する場合、MI＝1で、STRIを行ない、データをメモリ・バスに書きもどす。
(15)	メモリ・バス CI	メモリ・バスのCIが、Dirty Singleのエントリにヒットした場合、システム・バスにSINを行ない、エントリを無効にする。
	メモリ・バス CRI	メモリ・バスのCRIが、Dirty Singleのエントリにヒットした場合、OWN＊を出力し、システム・バスSTRIで得たデータをメモリ・バスに返し、エントリを無効にする。
	ディレクトリ リプレースメント	Dirty Singleのエントリを無効化する場合、STRIを行って、データをメモリ・バスに書きもどす。

【0136】以上のような操作を行うことによって、メモリ・バス21上のみのスヌープ動作と同様に、キャッシュ・メモリ23上でデータが行進されるときに唯一のキャッシュ・メモリ23にのみそのデータが存在することが保証され、キャッシュ・メモリ23間でのデータの不一致が発生しない。これを図3に示したキャッシュ・メモリの状態遷移において、説明する。外部メモリ・バス21へのCR時には、必ずSHR＊が出力されるので、（1）の遷移は発生せず、必ず（2）の遷移となる。従って、キャッシュ・メモリ23上での更新を検知することのできない（4）の遷移は発生しないことになり、必ず更新時に唯一のデータであることが保証されるためである。

【0137】なお、実施例ではライト・インバリデート（Write Invalidate）方式のスヌープ機構を持ったメモリ・バスを用いたが、ライト・ブロードキャスト（Write Broadcast ）方式等、他のスヌープ機構にも本発明を適用することが可能である。この場合には、システム・バス上のプロトコル等に変更が必要であるが、外部キャッシュ・メモリに存在しているデータのディレクトリを管理して、キャッシュ・メモリ上のデータの不一致の発生を避けるという基本構成に変わりはない。

【0138】また、実施例では64ビット幅のパラレル・バスを用いることにしたが、シリアル・バス等のように全く異なったバス構成に対しても本発明を適用することができる。更に、実施例ではキャッシュ・メモリを介してCPUをメモリ・バスに接続したが、プロセッサ・セグメント内のキャッシュ・メモリの配置についてはこ

35

れに限るものではない。これらについての変形例を次に説明する。

【0139】変形例

【0140】図9は、本発明の第1の変形例におけるデータ処理装置の構成を表わしたものである。この第1の変形例では各CPU22A～22Cに1つずつ対応するキャッシュ・メモリは存在せず、各CPU22A～22Cはメモリ・バス21に直接接続されている。メモリ・バス21にはこれらのCPU22A～22Cに共通するキャッシュ・メモリ23Dが配置されている。各プロセッサ・セグメント12₁～12ₙはこのような変更点を有している。それぞれのプロセッサ・セグメント12₁～12ₙ内のバス・インターフェース25₁～25ₙには、自己のメモリ・バス21₁～21ₙ上のメモリ24₁～24ₙ内のデータが外部のキャッシュ・メモリ内に保持されていることを記憶するためのディレクトリ26₁～26ₙが接続されているのは先の実施例と同様である。

【0141】図10は、本発明の第2の変形例におけるデータ処理装置の構成を表わしたものである。この第2の変形例でも各CPU22A～22Cに1つずつ対応するキャッシュ・メモリは存在せず、各CPU22A～22Cはメモリ・バス21に直接接続されている。メモリ24は、これらのCPU22A～22Cに共通するキャッシュ・メモリ23Eを介してメモリ・バス21に接続されている。各プロセッサ・セグメント12₁～12ₙはこのような変更点を有している。それぞれのプロセッサ・セグメント12₁～12ₙ内のバス・インターフェース25₁～25ₙには、自己のメモリ・バス21₁～21ₙ上のメモリ24₁～24ₙ内のデータが外部のキャッシュ・メモリ内に保持されていることを記憶するためのディレクトリ26₁～26ₙが接続されているのは先の実施例および第1の変形例と同様である。

【0142】

【発明の効果】このように本発明によれば、データ処理装置を複数のプロセッサ・セグメントとこれらを接続するシステム・バスで構成したので、それぞれのプロセッサ・セグメント内における各メモリ・バス上にローカルに存在するデータについては、システム・バスの介在なしに処理することができ、高速なアクセスが可能である。したがって、大規模なマルチ・プロセッサを構成した場合や、速度の遅いシステム・バスを採用した場合でも、ソフトウェアの最適化等の手法によってデータ処理装置の性能低下を最小限に抑えることができるという効果がある。

36

【図面の簡単な説明】

【図1】 本実施例におけるデータ処理装置の構成を表わしたブロック図である。

【図2】 本実施例のメモリ・バスの構成によるバス・アクセスのシーケンスを表わしたタイミング図である。

【図3】 キャッシュ・エントリがCPUおよびメモリ・バスの動作に対して取るべき状態遷移を示した説明図である。

【図4】 本実施例でシステム・バス信号を用いて行われるバス・トランザクション・シーケンスを表わした説明図である。

【図5】 本実施例でシステム・バス信号を用いて行われるバス・トランザクション・シーケンスを表わした説明図である。

【図6】 本実施例におけるアービトレーション回路の構成を表わした回路図である。

【図7】 本実施例における一連のアービトレーション・シーケンスを表わしたタイミング図である。

【図8】 本実施例におけるディレクトリの1エントリの状態遷移を表わした説明図である。

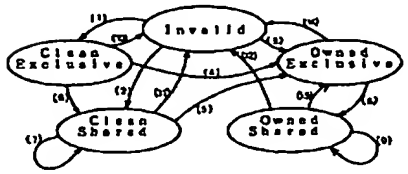
【図9】 本発明の第1の変形例におけるデータ処理装置の構成を表わしたブロック図である。

【図10】 本発明の第2の変形例におけるデータ処理装置の構成を表わしたブロック図である。

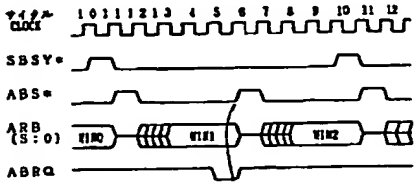
【符号の説明】

11…システム・バス、12…プロセッサ・セグメント、21…メモリ・バス、22…CPU、23…キャッシュ・メモリ、24…メモリ、25…バス・インターフェース

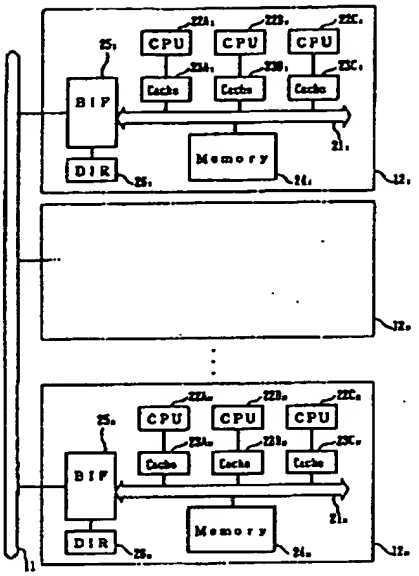
【図3】



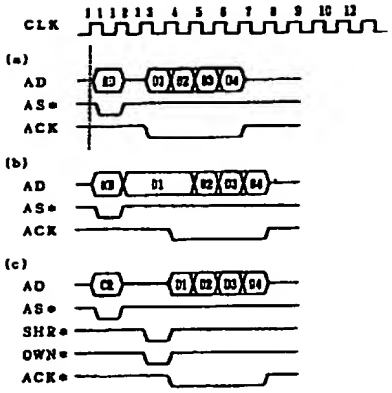
【図7】



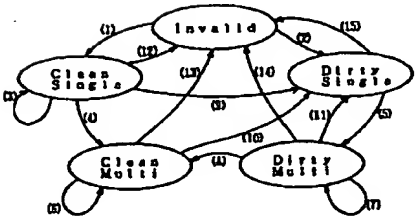
【図1】


CPU
CPU
CPU
BIF
Cache
Cache
Cache
Memory
DIR

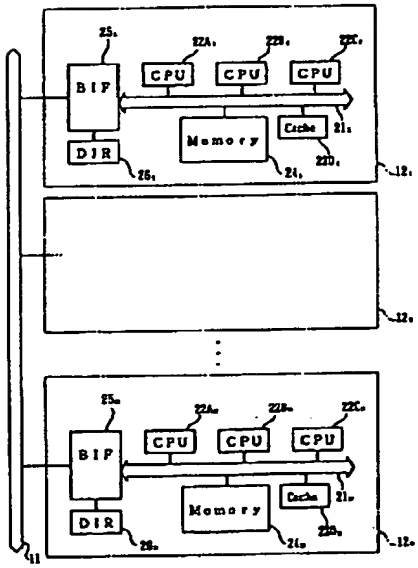
【図2】


CLK
(a)
AD
AS*
ACK
(b)
AD
AS*
ACK
(c)
AD
AS*
SHR*
OWN*
ACK*

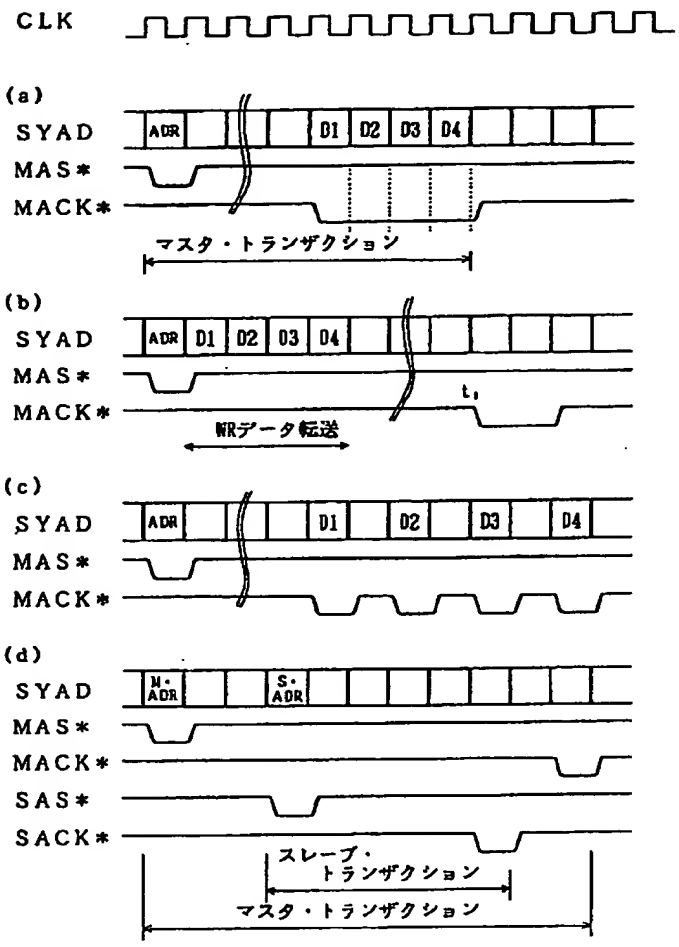
【図8】


Invalid
Clean Single
Dirty Single
Clean Multi
Dirty Multi

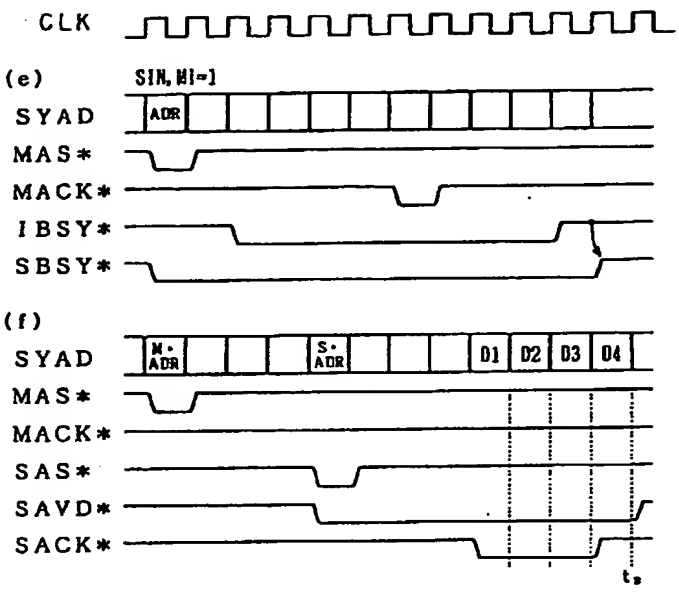
【図9】


BIF
CPU
CPU
CPU
Memory
Cache
DIR

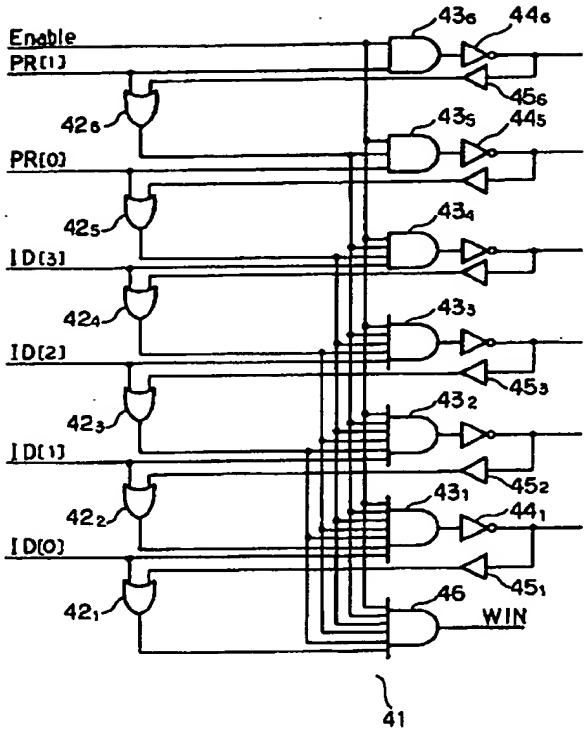
【図4】


CLK
(a)
SYAD
ADR
D1
D2
D3
D4
MAS*
MACK*
マスタ・トランザクション
(b)
SYAD
ADR
D1
D2
D3
D4
MAS*
MACK*
t1
WRデータ転送
(c)
SYAD
ADR
D1
D2
D3
D4
MAS*
MACK*
(d)
SYAD
M・ADR
S・ADR
MAS*
MACK*
SAS*
SACK*
スレーブ・トランザクション
マスタ・トランザクション

【図5】


CLK
(e)
SIN, MI=1
SYAD
ADR
MAS*
MACK*
IBSY*
SBSY*
(f)
SYAD
M. ADR
S. ADR
D1
D2
D3
D4
MAS*
MACK*
SAS*
SAVD*
SACK*
t_s

【図6】


Enable
PR[1]
PR[0]
ID[3]
ID[2]
ID[1]
ID[0]
42_6
42_5
42_4
42_3
42_2
42_1
43_6
43_5
43_4
43_3
43_2
43_1
44_6
44_5
44_1
45_6
45_3
45_2
45_1
46
WIN
41

【図１０】